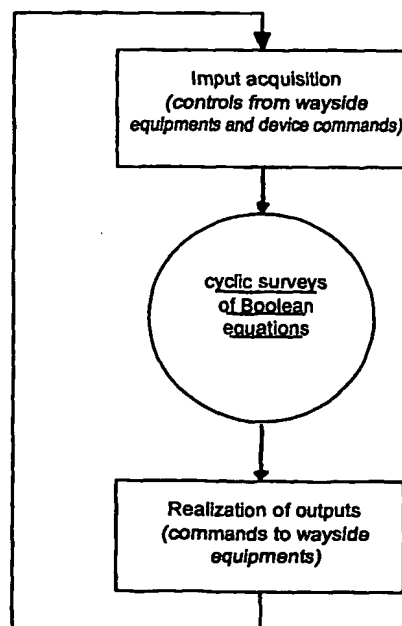
*Fig. 2*

INTERNAL FUNCTIONS OF SIMBOOL

*Fig. 3*

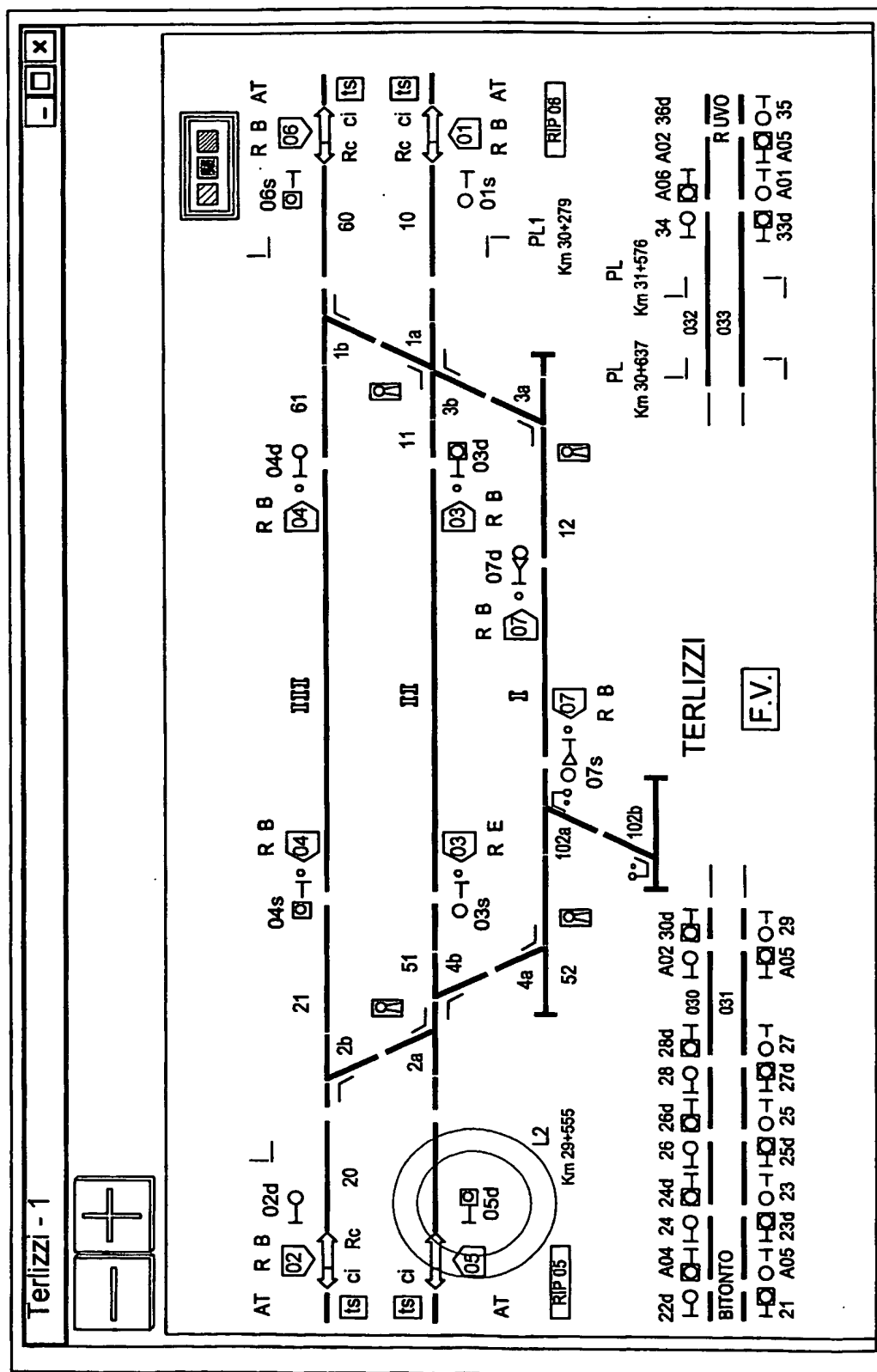


Fig. 4

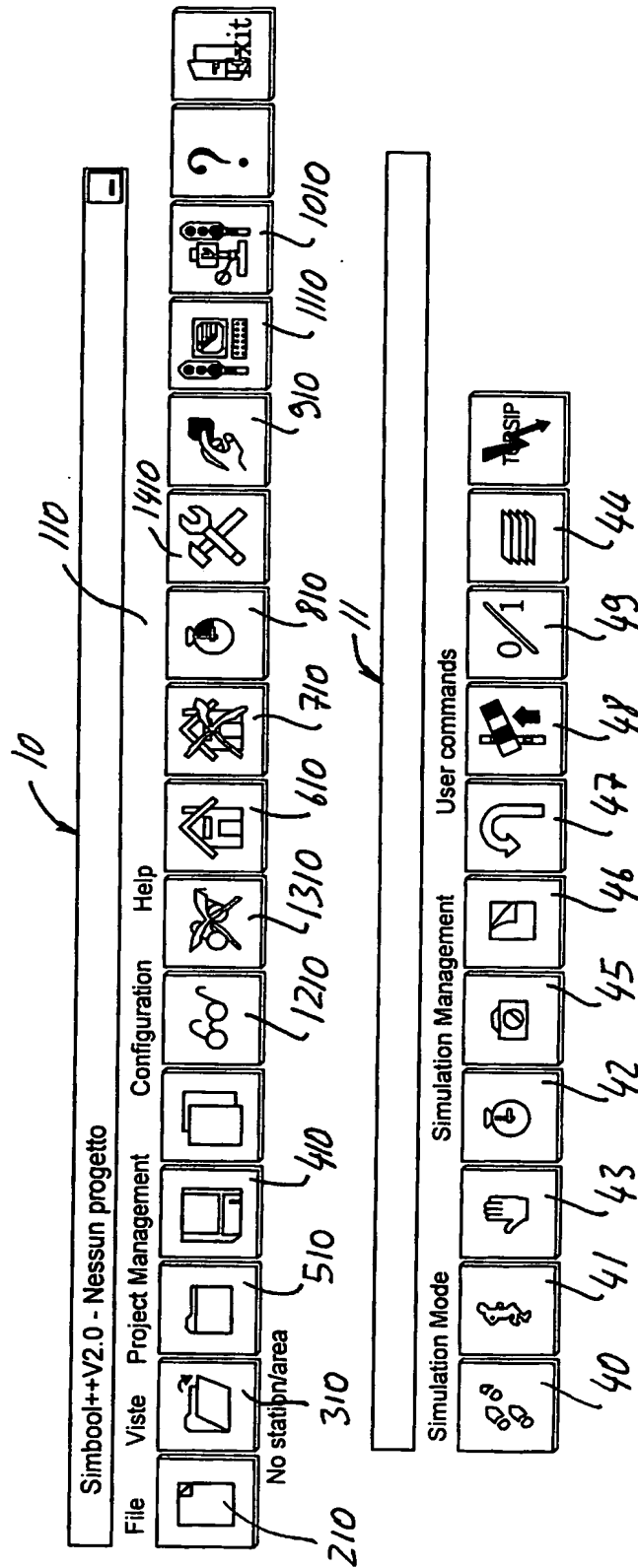
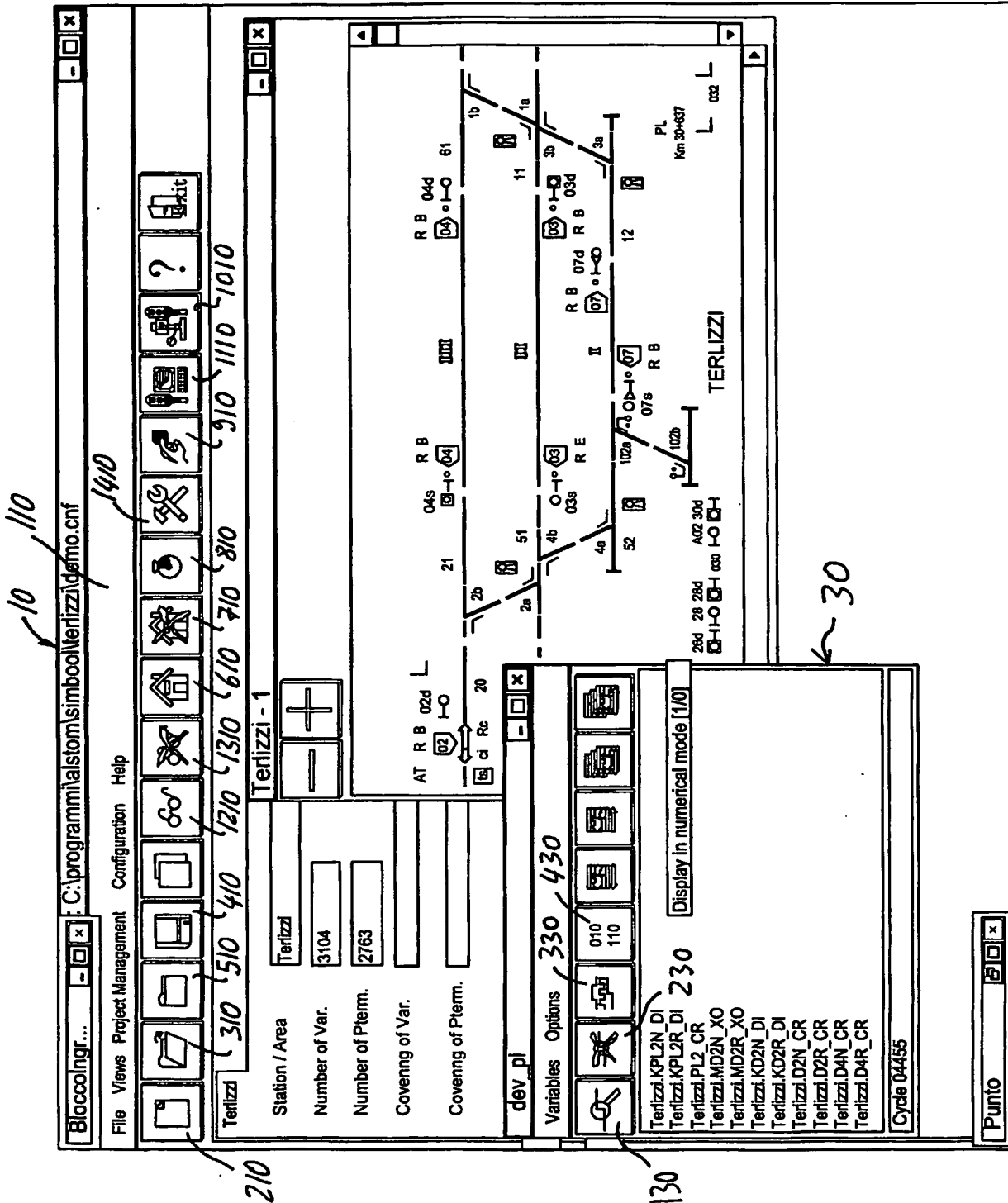


Fig. 5

10/531960

Fig. 6



Equipment simulator

Type

Add

Modify

Cancel

Close

AnnouncementDistantTrain
AnnouncementNearTrain
B of Entry Approach
B of Departure Approach
Reversible entry block
Reversible departure block
cdb
CMD AI T1 A INV

Fig. 7

CMD

Label

Alias

Type

Input / Output

Add

Modify

Cancel

Close

CM_M_TLD_CB101_PLS	PLS	Parameter	Output
TLD	TLD	Check	Input / Output
CONT	CNT	Local state	Input / Output

Fig. 8

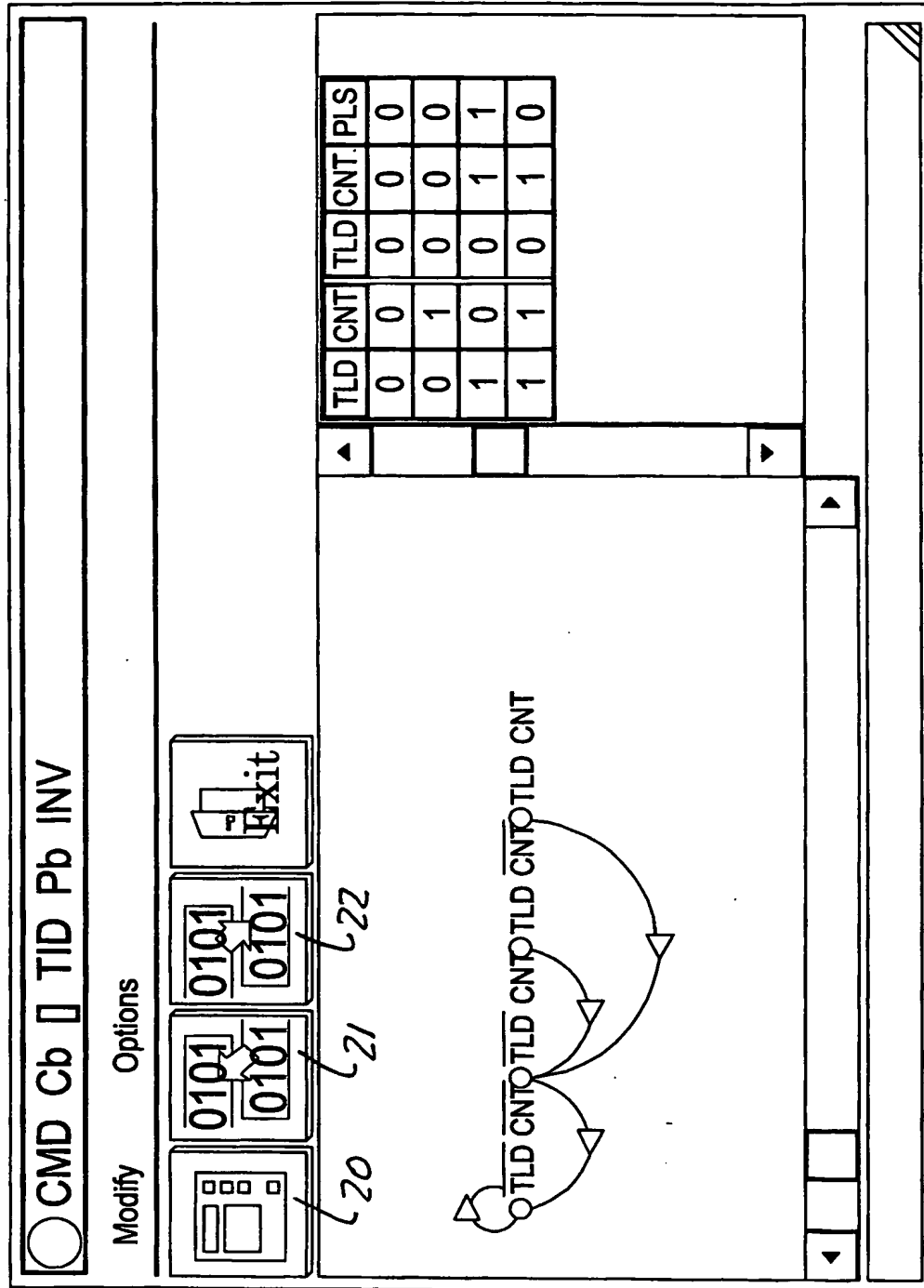


Fig. 9

* It is a cell value not initialized. If the table includes "*" the simulator logic is not generated and associated to any variables.

X If this value is assigned to a cell, even all the cells on the right side will be "X" for that row. This means that the state identified by the corresponding row is not allowed. Practically it is a combination not allowed or not used during the simulation.

TLD	CNT	TLD	CNT	PLS
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	0	1	0

0 This means that, when the input conditions in this cycle are checked, the output in the next cycle will go at "0".

1 This means that, when the input conditions in this cycle are checked, the output in the next cycle will go at "1".

Fig. 10

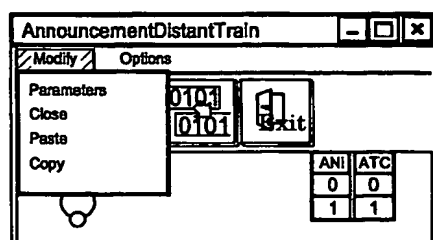


Fig. 11

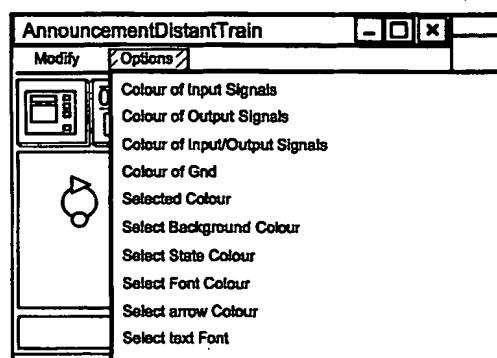


Fig. 12

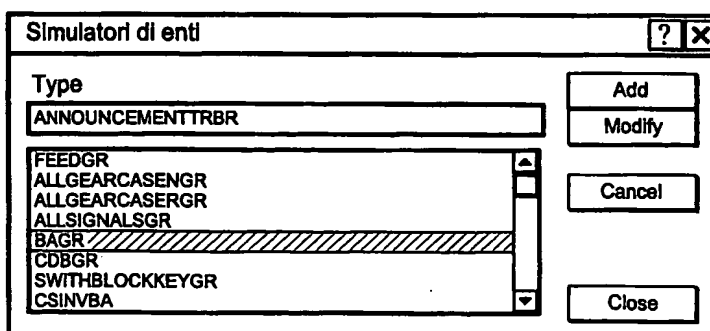


Fig. 13

10/18/01/18

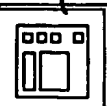
FEEDGR						-	□	x
Aggregates Selections								
<div>  20 </div>								
LP_I01_B_LOXMT	LP_I01_B_FLS	LP_I01_F_LOXMT	LP_I01_F_FLS					
0	0	0	0			<input type="checkbox"/>		Warning and TT absent
1	0	0	0					No State
0	1	0	0					Warning absent and TT present
0	0	1	0			<input checked="" type="checkbox"/>		Warning and TT present
0	0	0	1			<input checked="" type="checkbox"/>		Warning present and TT absent

Fig. 16

Condition Table

Path for Condition Table

C:\programs\alstom\simboo\terizzi\Tdo.nap.ini

Search

Confirm Cancel

Fig. 17

Condition Table	block_list
block_list	block block_list block
block	block_head line_list
block_head	[identifier]
line_list	line line_list line
line :	identifier = identifier_list
identifier_list :	identifier identifier_list identifier
identifier :	[A-Za-z0-9]

Fig. 18

(ASCV)
NAME = AREA
AREA_TE = AREA 110 AREA 371
CDB_STAT = CB 129 CB 137 CB 130 CB 131 CB 132 CB 128
CDB_LINE = CBL61 CBL59
DEV_SEM = DV 107 DV 109 DV 110 DV 111 DV 112 DV 105
DEV_COM = DV 103 DV 108
RAUT = DV 107 DV 109
SA = SA 141 SA 142 SA 143 SA 144 SA 145 SA 146 SA 147
SB = SB 103 SB 105 SB 104 SB 132 SB 131 SB 130
PTM = PTM 103 PTM 104 PTM 105 PTM 121 PTM 122
PT = PT 121 PT 141 PT 142 PT 143 PT 144 PT 145 PT 146 PT 147 PT 148 PT 149 PT 150 PT 151 PT 152 PT 122 PT 123 PT 124
PT 125 PT 126 PT 127 PT 128 PT 129 PT 130 PT 131 PT 132 PT 11 PT 10
RAR = RAR1
CNTR = 1_EPS_P SERV_N_ACS1_V
CMD_TO = COMM_N_ACS1 COMM_R_ACS1

(ROUTING)
NAME = IS 104_146
PTO = PTM 104
PTF = PTM 146
SBO = SB 104
SBF = SB 146
OPZ = IS 104_1461

(OPTION_IS)
NAME = IS 104_1461
DIR = RIGHT
ULT = CB 128
LIB = CB 104
CDB = CB 105
DEVS = DV 104 OPPOSITE DV 105 OPPOSITE
ZTE = AREA 110
SB = SB 128
INC_IS = IS 128_1021 IS 128_1111 IS 104_1451 IS 104_1441 IS 125_1021 IS 125_1111 IS 124_1021 IS 124_1111 IS 191_1261
INC_IT = IT 125_111 IT 124_111 IT 10_1451 IT 10_1441 IT 126_111 IT 10_1461
CDB_LIB = CB 105
CMD_TO = CMD 104_146
VAR_TF = IS 104_146

Fig. 19

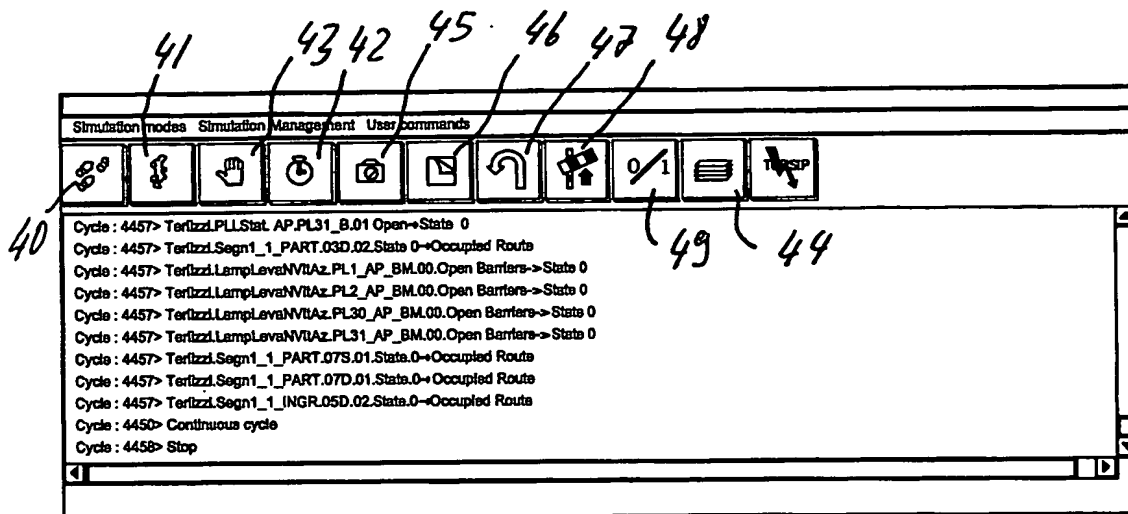


Fig. 20

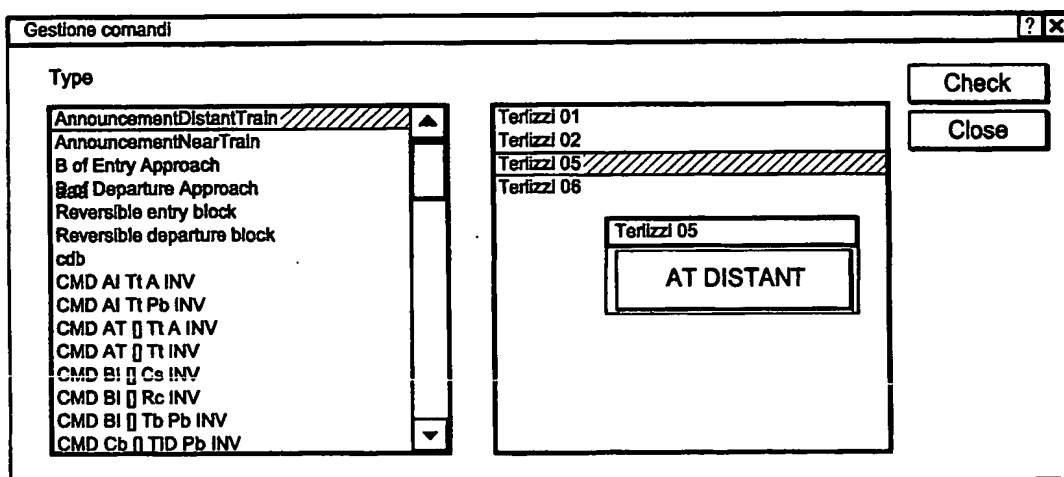


Fig. 21

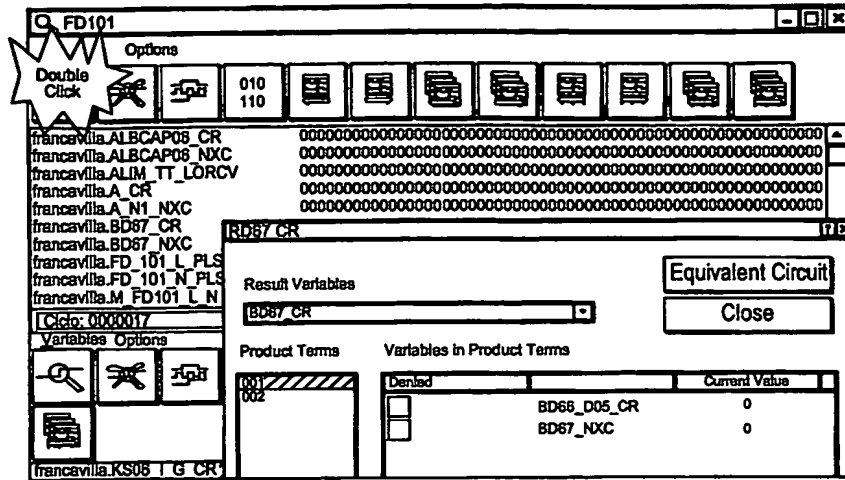


Fig. 23

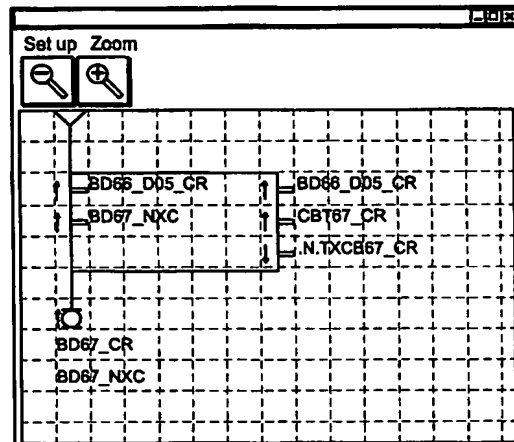


Fig. 24

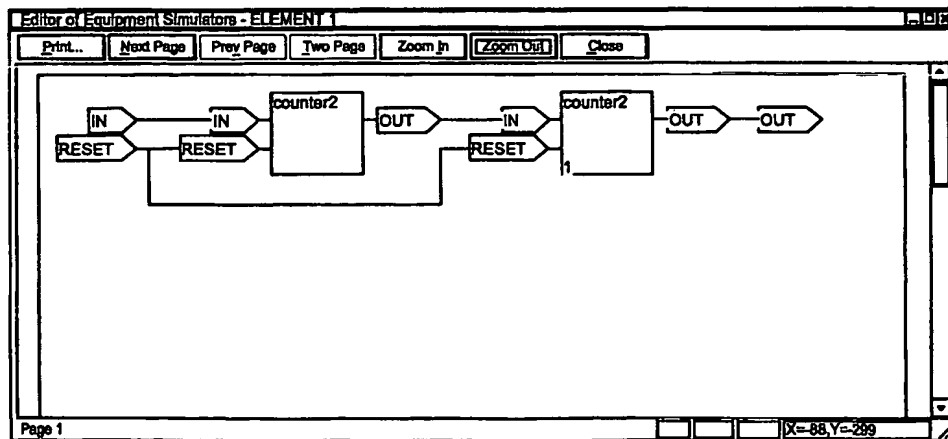


Fig. 25

ENCLOSURE PAGE A1

REPORT FILE EXECUTING TEST

```

proc Main { Istr } {
    global Error

    global Name
    global codResult1
    global codResult2
    global Step
    global ProgrAcc

    global ListDevsd
    global ListDevc
    global CompleteListCdb

    global FlgRr

    set ls [string length $Istr]
    set FlgOpz [string first "_OPZ" $Istr]

    if { $FlgOpz > 0 } {
        set ld [expr $ls - 5]
    } else {
        set ld [expr $ls - 2]
    }

    set IstrNoOpz [string range $Istr 0 $ld]

    # ----- #
    #                                     #
    #   READING DATA BASE ROUTING      #
    #                                     #
    # ----- #

    # Reading the name starting pto
    set Pto [ Read CFG CTDC 0 IST $IstrNoOpz PTO ]

    # Reading the name of ending point of the routing
    set Ptf [Read CFG CTDC 0 IST $IstrNoOpz PTF ]

    # Reading the name of SB of the starting point of the routing
    set Sbo [Read CFG CTDC 0 IST $IstrNoOpz SBO ]

    # Reading the name of SB of the ending point of the routing
    set Sbf [Read CFG CTDC 0 IST $IstrNoOpz SBF ]

    # Reading if the starting point of the rout. requires the cdb occupation
    # (~existing of 'train presence on cdb' )
    set LibPto [Read CFG CTDC 0 PTM $Pto CLEARING ]

    # Reading the simple switch blocks of the routing
    set ListDevs [Read CFG CTDC 0 IST $Istr DEVS ALL ]

    # Reading the switch blocks for communicating the routing
    set ListDevc [Read CFG CTDC 0 IST $Istr DEVC ALL ]

    # Reading buffer stops
    set ListSfc [Read CFG CTDC 0 IST $Istr SFC ALL ]

    # Creating list of cdb to be covered to execute the train transit
    set CompleteCdbList [ CreateCompleteCdbList $Istr ]

    set Area [ GetAreaIstr $Pto ]

```

ENCLOSURE PAGE A2

REPORT FILE EXECUTING TEST1

```
set DataEqu [ ReadCFG CTDC 0 ASCV $Area DATA_
set DataTdc [ ReadCFG CTDC 0 ASCV $Area DATA_
```

```
if { [ llenght $DataEqu ] > 0 } {
    Message2 "Data Equations : $DataEqu"
```

```
if { [ llenght $DataTdc ] > 0 } {
    Message2 "Data tdc : $DataTdc"
    Message2 " "
}
```

```
# ----- #
#           #
#   STEF 1   #
#           #
# ----- #
```

```
### Blocking switch points in opposite position than the TDC
#
Message " Blocking switch points in clashing position than the TDC "
# Command levers in clashing position
PositionClashingList $Istr $ListDevs DEVS
# Command levers in clashing position
PositionClashingList $Istr $ListDevc DEVC
# Command in clashing position
PositionClashingList $Istr $ListSfc SFC
#
###
```

```
set ProgrAcq 1
set FlgRr "F"
### Checking if switch points are at state : clashing FREE CONTROL
#
Message "Check switch points in state < clashing FREE CONTROL >"
# Checking clashing position
CheckPositionClashingList $Istr $ListDevs DEVS FREE CONTROL
CheckPositionClashingList $Istr $ListDevc DEVC FREE CONTROL
CheckPositionClashingList $Istr $ListSfc SFC FREE CONTROL
#
###
```

```
Message "Commanding Routing $Istr"
```

```
if { $FlgOpz > 0 } {
    Command TF 0 IST $IstrNoOpz COMMAND OPZ
} else {
    Command TF 0 IST $IstrNoOpz COMMAND ""
}
```

```
# Check current state of the routing.
set Name $Istr
set codResult1 "1"
set codResult2 "2"
set Step 1
set ProgrAcq 1
set FlgRr "T"
CheckStateIstr REST TRUE RECORDED FALSE "" "" $Pto $Ptf
```

```
# Getting the name of the last switch point of the routing
set ListDevIstr [ concat $ListDevs $ListDevc $ListSfc ]
set NameLastDev [ lindex $ListDevIstr [ expr [llenght $ListDevIstr] - 1] ]
```

```
### Loop for all the switch points
```

```
#
set Step 2
set ProgrAcq 0
```

```
if { [ llenght $ListDevs ] > 0 } {
    foreach NameDev $ListDevs {
        # Commanding switch point in central position (AUTOMATIC)
        Message "Commanding switch point $NameDev in AUTOMATIC position"
        Command TF 0 DEV $NameDev AUTOMATIC ""

        Message "Commanding Routing $Istr"
        if { $FlgOpz > 0 } {
            Command TF 0 IST $IstrNoOpz COMMAND OPZ
        }
    }
}
```


ENCLOSURE PAGE A3
REPORT FILE EXECUTING TEST

```

    } else {
        Command TF 0 IST $IstrNoOpz COMMAND ""
    }

    set ProgrAcq [expr $ProgrAcq + 1 ]
    if { $NameDev == $NameLastDev } {
        Message "Checking if the switch is in position < concordant BLOCKED CONTROL > "
        CheckSwitchPointState $Istr DEVS $NameDev CONCORDANT BLOCKED CONTROL
    } else {
        Message "Checking if the switch is in position < concordant FREE CONTROL > "
        CheckSwitchPointState $Istr DEVS $NameDev CONCORDANT FREE CONTROL
    }

    set ProgrAcq [expr $ProgrAcq + 1 ]
    Message "Checking routing state"
    # Check current routing state
    # Controlling if it is the last one
    if { $NameDev == $NameLastDev } {
        set codResult1 "3"
        set codResult2 "4"
        CheckIstrState PO_BLOCKED TRUE RECORDED TRUE "" "" $Pto $Ptf
    } else {
        set codResult1 "1"
        set codResult2 "2"
        CheckIstrState REST TRUE RECORDED FALSE "" "" $Pto $Ptf
    }
}
}
set tmpDev ""

if { [ llength $ListDevs ] > 0 } {
    foreach NameDev $ListDevs {
        if { $tmpDev = $NameDev } {
            # Commanding switch point in central position (AUTOMATIC)
            Message "Commanding switch point $NameDev in AUTOMATIC position"
            Command TF 0 DEV $NameDev AUTOMATIC ""

            Message "Commanding Routing $Istr"
            if { $FlgOpz > 0 } {
                Command TF 0 IST $IstrNoOpz COMMAND OPZ
            } else {
                Command TF 0 IST $IstrNoOpz COMMAND ""
            }

            set ProgrAcq [expr $ProgrAcq + 1 ]
            if { $NameDev == $NameLastDev } {
                Message "Checking if the switch is in position < concordant BLOCKED CONTROL > "
                CheckSwitchPointState $Istr DEVC $NameDev CONCORDANT BLOCKED CONTROL
            } else {
                Message "Checking if the switch is in position < concordant FREE CONTROL > "
                CheckSwitchPointState $Istr DEVC $NameDev CONCORDANT FREE CONTROL
            }

            set ProgrAcq [expr $ProgrAcq + 1 ]
            Message "Checking routing state"
            # Check current routing state
            # Controlling if it is the last one
            if { $NameDev == $NameLastDev } {
                set codResult1 "3"
                set codResult2 "4"
                CheckIstrState PO_BLOCKED TRUE RECORDED TRUE "" "" $Pto $Ptf
            } else {
                set codResult1 "1"
                set codResult2 "2"
                CheckIstrState REST TRUE RECORDED FALSE "" "" $Pto $Ptf
            }
            set tmpDev $NameDev
        }
    }
}
set codResult1 "10"
set ProgrAcq [expr $ProgrAcq + 1 ]

```

ENCLOSURE PAGE A4
REPORT FILE EXECUTING TEST

```
### Checking low signal state starting point
#
Message " Checking low signal state starting point "
if { [ check DAS 0 SB $Sbo FREEROUTE TRUE 3 0 ] == "TRUE" } {
    Message " CHECK OK : The low signal $Sbo is FREE TRANSIT"
    PrintRealRisults $Name $Sbo $codResult1 $Step $ProgrAcq
} else {
    set error "1"
    Message " CHECK FAILED " : The low state $Sbo is not FREE TRANSIT"
    if { $FlgRr == "T" } {
        set ECodResult "E"
        append ECodResult $codResult1
        PrintRealRisults $Name $Sbo $codResult $Step $ProgrAcq
    }
}
#

# ----- #
#                               #
#     STEP 2                     #
#                               #
# ----- #
```